

**NJIT**

New Jersey's Science &  
Technology University

*THE EDGE IN KNOWLEDGE*

**CS 280**  
**Programming Language**  
**Concepts**

**Strings**

## What's a string?

- A sequence of characters that are grouped together and dealt with as a single unit
- Different languages handle strings in different ways, and provide tools to help you deal with strings in that language

## OK, so first: what's a character?

- In natural languages, a character is a graphical symbol (a letter, a number, a punctuation mark, etc).
- In computing we represent a character as a pattern of bits. We usually keep it in a variable of type `char`.
- This is often an 8 bit quantity.
- By convention, the ASCII encoding for a character is used
- Usually, a character constant is a single character enclosed in single quotes. The sequence `'a'` is a character constant for the bit pattern that represents the letter a in ASCII
- NOTE: more modern languages use a form of "Unicode" encoding, but an 8-bit "char" predates UniCode and is deeply ingrained into C-like languages
- In Java, a char holds a 16 bit unicode character

## Other ways of representing chars

- You can also put an octal or hex value inside the single quotes, which represents that bit pattern in a character
  - 'a' and '\141' and '\x61' are all the same thing
- There are so-called escape sequences which represent common characters:
  - '\n' newline
  - '\t' tab
  - '\"' a single quote character
  - '\\' a backslash character
  - '\0' the null character (all zero bits)

## Strings in C

- A string in C (a "C-string") is a sequence of characters with a null character included ('\0') as an end marker
- A C-string is thus nothing more than an array of characters with an end marker
- Note that there is no "string type"; instead there's just the convention of marking the end of an array of chars with a null char ('\0') and calling it a string
- C has a built in shorthand for these arrays
- There are library routines that follow the convention that a string is an array ending in '\0'

## C shorthand for strings

- This is a string:  

```
{ 'h', 'e', 'l', 'p', ' ', 'm', 'e', ' ', 'r', 'h', 'o', 'n', 'd', 'a', '!', '\0' }
```

//that's an array initialization
- So is this:  

```
"help me rhonda!"
```
- A double-quoted list of characters is a shorthand for a string in C
- The compiler automatically adds the null character for you at the end of a quoted string

## Variables for strings

- A string is an array of characters, so...
  - `char mystring[7] = { 'h', 'e', 'l', 'l', 'o', '!', '\0' };`
  - `char mystring[7] = "hello!";`
  - `char mystring[] = { 'h', 'e', 'l', 'l', 'o', '!', '\0' };`
  - `char mystring[] = "hello!";`
- All of these are a declaration of an array of 7 characters, initialized to the string hello!

## C String Libraries

- The C standard library has functions that follow the “array of characters ending with a null” convention
- For example
  - `strlen(s)` returns the length of the string in `s`
  - `strcpy(to, from)` copies the string in `from` into the string in `to`
  - `printf` is used to generate formatted output
- Notice that a C-string is not an object, so these are not “method calls” against a string. Instead, you must pass the string to the library function

## Some examples

```
char onestr[] = "hello!";
char another[] = "there";
char athird[100];

// print the length of onestr
printf("%d\n", strlen(onestr) );

// copy from one array into another
strcpy(athird, onestr);

// print strings
printf("%s\n", onestr);
printf("%s\n", athird);

cout << onestr; // this is how we print a C string in C++
```

## Error Examples

What if I tried to strcpy into something that isn't big enough to fit what I'm trying to copy?

- bad things. very bad things.
- C requires you to manage your own memory

```
onestr = another;           // << NO!!!!
```

- this is invalid C and will not compile: C strings are NOT types, you can't assign them
- both onestr and another are the names of arrays. You can NOT assign to the name of an array

```
strcpy(another, onestr);    // << NO!!!!
```

- this might break. on some systems, constants are read only

## Additional string functions

- strncpy – strcpy with a length parameter
- strcat – concatenate strings
- strcmp – compare strings
- strncmp – compare strings, with a length parameter

## Assessment of C strings

- It's pretty simple: all the string functions just deal with arrays of characters
- Not very efficient in some cases (for example, `strlen` looks at every character in the string)
- Might not be very safe

## Strings in C++

- Just about everything in C is in C++, therefore everything we've discussed about strings in C is true for C++
- C++ ALSO has a definition for a string class; basically a definition of a type for strings.

## Example

```
#include <iostream>
#include <string>
using namespace std;

int
main()
{
    string mystring = "hello!";
    string yourstring = mystring;

    cout << mystring.length() << '\n';

    return 0;
}
```

## Observations

- A C++ string (a.k.a. `std::string`) can be initialized from a “quoted string” (a.k.a. a C string)
- You can initialize one string from another using assignment
- The string keeps track of and manages memory
- Length does not require looking at the entire string to count characters
- You can concatenate strings with `+` because the string class overloads the addition operator for strings



## More string stuff

- The relational operators work on C++ strings
  - `string1 == string2` is valid in C++ strings
- The i-th character in a C string is referenced using array operations: `mystring[i]` is the i-th character
- That works on a C++ string, too, because the library overloads the `[]` operator
- There's also an “at” function that does the same thing for C++ strings: `mystring.at(i)`
- Similar to Java String “`charAt()`” method
- The convention in C++ is that `at()` does bounds checking and `[]` does not

## And in Java?

- There is a String object (`java.lang.String`) defined
- Similar things can be done in Java: initialize from a “quoted string”, copy values
- The names of the string functions are different
- You can concatenate strings with `+`
- You can not compare strings with `==`

## Concluding observations

- Three approaches to the same problem
- Two approaches appear in C++
- The concepts are similar but the implementations are different
- Annoyingly, the function names are different
- Observations we will come back to:
  - C++ has a way to make operators work on objects
  - Memory is important, and managing it is up to you